Towards Automatic Partitioning of Class Hierarchies

Stuckenschmidt, Heiner; Klein, Michel

published in
Proceedings of the 1st International Conference on Knowledge Management and Decision Support (ICKMDS'04)
2004

document version
Peer reviewed version

document license
Unspecified

Towards Automatic Partitioning of Class Hierarchies

Heiner Stuckenschmidt
KR & R Group
Vrije Universiteit Amsterdam
Amsterdam, Netherlands
heiner@cs.vu.nl

Michel Klein
KR & R Group
Vrije Universiteit Amsterdam
Amsterdam, Netherlands
michel.klein@cs.vu.nl

***Abstract*—The increasing awareness of the benefits of ontologies for information processing has lead to the creation of a number of large ontologies about real world domains. The size of these ontologies and their monolithic character cause serious problems in handling them. In other areas, e.g. software engineering, these problems are tackled by partitioning monolithic entities into sets of meaningful and mostly self-contained modules. In this paper, we suggest a similar approach for ontologies. We propose propose an approach for automatically partitioning large ontologies into smaller modules based on the structure of the class hierarchy. The method is demonstrated on a part of the UMLS semantic network. Experiments with larger ontologies are available online at** `http://swserver.cs.vu.nl/partitioning/`

Keywords: Ontologies, Modularization, Graph Analysis

I. INTRODUCTION

The increasing awareness of the benefits of ontologies for information processing in open and weakly structured environments has lead to the creation of a number of such ontologies for real world domains. In complex domains such as medicine these ontologies can contain thousands of concepts. Examples of such large ontologies are the NCI cancer ontology [5] with about 27.500 and the Gene ontology [7] with about 22.000 concepts. Other examples can be found in the area of e-commerce where product classification such as the UNSPSC or the NAICS contain thousands of product categories. While being useful for many applications, the size and the monolithic nature of these ontologies causes new problems that affect different steps of the ontology life cycle.

Maintenance Ontologies that contain thousands of concepts cannot be created and maintained by a single person. The broad coverage of such large ontologies normally requires a team of experts. In many cases these experts will be located in different organizations and will work on the same ontology in parallel. An example for such a situation is the gene ontology that is maintained by a consortium of experts.

Publication Large ontologies are mostly created to provide a standard model of a domain to be used by developers of individual solutions within that domain. While existing large ontologies often try cover a complete domain, the providers of individual solutions are often only interested in a specific part of the overall domain. The UNSPSC classification for example contains categories for all kinds of products and services while the developers of an online computer shop will only be interested in those categories related to computer hardware and software.

Validation The nature of ontologies as reference models for a domain require a high degree of quality of the respective model. Representing a consensus model, it is also important to have proposed models validated by different experts. In the case of large ontologies it is often difficult if not impossible to understand the model as a whole due to cognitive limits. What is missing is an abstracted view on the overall model and its structure as well as the possibility to focus the inspection of a specific aspect.

Processing On a technical level, very large ontologies cause serious scalability problems. The complexity of reasoning about ontologies is well known to be critical even for smaller ontologies. In the presence of ontologies like the NCI cancer ontology, not only reasoning engines but also modelling and visualization tools reach their limits. Currently, there is no OWL-based modelling tool that can provide convenient modelling support for ontologies of the size of the NCI ontology [5].

All these problems are a result of the fact that the ontology as a whole is too large to handle. Most problems would disappear if the overall model consists of a set of coherent modules about a certain subtopic that can be used independently of the other modules while still containing information about its relation to these other modules.

- In distributed development, experts could be responsible for an single module and maintain it independently of other modules thus avoiding revision problems.

- Users of an ontology could use a subset of the overall ontology by selecting a set of relevant modules. While only having to deal with this relevant part, the relations to other part of the model is still available through the global structure.

- Validation of a large ontologies could be done based on single modules that are easier to understand. Being related to a certain subtopic, it will be easier to judge the completeness and consistency of the model. Validated modules could be published early while other parts of the ontology is still under development.

- The existence of modules will enable the use of software tools not able to handle the complete ontology. In the case of modelling and visualization tools, the different modules could be loaded one by one and processed individually. For reasoning tasks we could make use of parallel architectures where reasoners work on single modules and exchange partial results.

Recently, some proposals concerning the representation of modules and their connections have been made [3, 6, 8] that propose languages and discuss issues like the organization of modules and dependencies between them. A problem that has not been addressed yet concerns the creation of modules from existing ontologies. This problem we refer to as *modularization* or *partitioning* is discussed in the remainder of this paper.

The key question connected to modularization is about criteria for determining the assignment of concepts to modules. This requires a good intuition about the nature of a module. Intuitively, we can say that a module should contain information about a coherent subtopic that can be stand for itself. This requires that the concepts within a module are semantically connected to each other and do not semantically depend on information outside the module. These considerations imply the need for a notion of dependency between concepts that needs to be taken into account. There are many different ways in which concepts can be related explicitly or implicitly. At this point we abstract from specific kinds of dependencies and use a general notion of dependency between concepts. The resulting model of an ontology is the one of a weighted graph $O = \langle C, D, w \rangle$ where nodes C represent concepts and links D between concepts represent different kinds of dependencies that can be weighted according to the strength of the dependency. There are many different ways in which concepts can depend on each other. These dependencies can be reflected in the definitions of the ontology or can be implied by the intuitive understanding of concepts and background knowledge about the respective domain. Looking for an automatic partitioning method, we are only interested in such kinds of dependencies that can be derived from the ontology itself. This need leads us to the central assumption that dependencies between concepts can be derived form the structure of the ontology. Depending on the representation language, different structures can be used as indicators of dependencies. These structures can be subclass relations between classes, other relations linked to classes by the range and domain restrictions or the appearance of a class name in the definition of another class.

The paper is organized as follows. We first describe the basic steps of our proposed method which includes the creation of the dependency graph based on the ontology structure and the determination of modules. In section II we discuss the general approach and identify some problems. Based on the discussion we present an iterative algorithm in section III that assigns concepts to modules in such a way that minimal user input is required. We illustrate the algorithm using a part of the UMLS semantic network. We further discuss use cases for our partitioning method in section IV and conclude with an discussion of the proposed method and some directions for future improvements.

II. PARTITIONING METHOD

Our method consists of two tasks that are executed in four independent steps. The first task is the creation of a weighted graph from an ontology definition. This is done in two steps: extraction of the dependency structure and determination of the weight of the dependency. The second task concerns the identification of modules from the dependency graph. This task includes the detection of strongly related sets of concepts and the handling of unassigned concepts. In the following we discuss the techniques currently used in these steps.

A. Step 1: Create Ontology Graph

In the first step a dependency graph is extracted from an ontology source file. We implemented a PROLOG-based tool that reads OWL and RDF schema files using the SWI semantic web library [9] and outputs a graph format used by the networks analysis tool pajek [2] that we use for detecting related sets of nodes. The tool can be configured to use different kinds of dependencies. Currently it can extract dependencies corresponding to the subclass hierarchy and dependencies created by the domain and range restrictions in property definitions. Figure 1 shows the dependency graph created from an OWL version of the UMLS-semantic network using only the class hierarchy.

B. Step 2: Determine Strength of Relations

In the second step the strength of the dependencies between the concepts has to be determined. Following the basic assumption of our approach, we use the structure of the dependency graph to determine the weights of dependencies. In particular we use results from social network theory by computing the proportional strength network for the dependency graph. The proportional strength p_{ij} of a connection between a node c_i and c_j describes the importance of a link from one node to the other based on the number of connections a node has (a_{ij} is the weight preassigned to the link between c_i and c_j) [4]:

$$p_{ij} = \frac{a_{ij} + a_{ji}}{\sum_k a_{ik} + a_{ki}}$$

The intuition behind it is that individual social contacts become more important if there are only few of them. In our setting, this measure is useful because we want to present that classes that are only related to a low number of other classes get separated from them. This would be against the intuition that classes in a module should be related. Figure 2 shows the proportional strength network for the UMLS dependency graph.

C. Step 3: Determine Concept Islands

The proportional strength network provides us with a foundation for detecting sets of strongly related concepts. For this purpose, we make use of the 'island' algorithm [1]. A set of vertices $I \subseteq C$ is a line island in network if and only if it induces a connected subgraph and the lines inside the island are stronger related among them than with the neighboring vertices. In particular there is a spanning tree T over nodes in I such that

$$\max_{(u,v)\in V, v\notin T} w(u,v) < \min_{(u,v)\in T} w(u,v)$$

This criterion exactly coincides with our intuition about the nature of modules given in the introduction. The algorithm requires an upper and a lower bound on the size of the detected set as input. Figure 3 shows the result of determining islands of a size between 5 and 15 nodes.

D. Step 4: Assign Isolated Concepts

Depending on the nature of the dependency graph it may happen that some nodes cannot be assigned to an island. In Figure 3 there are 6 of these unassigned nodes marked with a zero (four concepts related to organizations as well as the concepts animal and invertebrate). As our definition of a partitioning does not allow unassigned classes, we have to assign these concepts to a modules as well. The example shows that leftover nodes can occur at different places in the graph and are not necessarily related. Therefore we chose assign them to existing modules. The assignment is based on the strength of the relation to nodes already assigned to a module. In particular leftover nodes are assigned to the island of a that neighboring node they have the strongest they have the strongest relation to. In cases where all neighboring nodes are unassigned as well, these nodes are assigned first. The concepts relating to organizations are assigned to module 2 (rooted at the general concept entity) and the concepts animal an invertebrate are assigned to module 7 (organisms).

E. Discussion

The different steps described above lead us to a partitioning of the input ontology into modules that satisfy the formal conditions mentioned in the previous section. One of the main problems with the approach as described above is the fact that we have to determine the size of modules that we want to be generated. the reason is that the optimal size of modules heavily depends on the size and the nature of the ontology. In some preliminary experiments we found a module size of one to ten per percent of the size of the complete ontology works quite well for some example ontologies that had between 1000 and 2000 concepts. This heuristic, however, did not work for larger or more fragmented ontologies. In some cases a bad choice of the upper and lower bound for the size of modules also led to an extremely high number of unassigned nodes that in turn created quite large modules with little internal coherence after reassigning them as described in step 4.

III. ASSIGNMENT STRATEGY

In order to avoid the problems caused by a wrong choice of upper and lower bond for the module size, we designed an algorithm that iterates over steps 3 and 4 of the process and generates 'natural' islands that are not influenced by the choice of a particular size. In the following we describe the algorithm and exemplify its effect using the same example as above.

A. Iterative Algorithm

The idea of the iterative assignment algorithm is to not prescribe the size of modules to be generated but to let them determined solely by the island criterion given in the last section. A way of doing this is to set the lower bound to 1 and the upper bound to $s - 1$ where s is the size of the complete ontology. Reducing the limit by one forces the algorithm to split up the ontology in some way as the complete model exceeds the upper size limit for an island. Choosing a limit that is just one below the size

of the complete ontology does not further restrict the selection of islands. This way we get the most natural grouping of concepts into strongly dependent sets. Even in this case where we do not restrict the size of island it can still happen, that nodes cannot be assigned to islands. Therefore we have to perform the extension step afterwards in order to assign these nodes to a module.

Algorithm 1 Partition

```
Require: limit: integer
Require: ontology: graph
Require: counter:integer
  current := ontology
  if |current| > limit then
    min := 1
    max := |current| − 1
    candidates := islands(min,max,current)
    for all module ∈ candidates do
      Expand(module,current)
      Partition(limit,module,counter)
    end for
  else
    counter := counter + 1
    for all c ∈ current do
      α(c) := counter
    end forreturn counter
  end if
```

As a result of this strategy the islands found by the algorithm can significantly differ in size. In particular, we often get large islands that cover most of the ontology. In order to get modules of a reasonable size, we iteratively apply the algorithm to islands that are too large to be useful modules. Often this applies only for a single large island, but there are also cases especially in the case of very large ontologies where the algorithm has to be applied recursively on different parts of the ontology. Nodes in islands that are small enough are assigned to a unique number and form a module of the ontology.

Algorithm 1 shows the corresponding labelling algorithm that takes a graph and labels the nodes of the graph with numbers that correspond to a partitioning assignment. The algorithm also needs the upper limit of the size of a module as input in order to determine when to stop the iterating. The counter is used in the recursive calls to make sure that modules have unique numbers When starting the iteration, the counter has to be set to zero.

B. Partitioning Example

We now illustrate the labelling algorithm described above using the UMLS model we also used to describe the different steps of our method. We chose an upper limit of 20 for the size of modules. For the relatively small example model, the algorithm only needs three iterations to determine modules. We discuss the result of the different iterations in the following.

Iteration 1 In the first iteration the algorithm already determines three islands that are smaller than 20 (compare Figure 4 . These islands consist of concepts related to biological active substances, different age groups as well as the subtree rooted at the concept concept or idea. While it could be argued that biologically active substances could be included in a larger module on substances the other two modules and in particular the one about ideas and concepts clearly contain concepts that are related and sufficiently different from the other concepts for form a module on their own.

Iteration 2 After removing the modules found in the first step iterating steps three and four results in another island of size lower than 20 namely the subtree rooted at the concept organism (compare Figure 5). This island is a good example of a very natural module found by the algorithm as the different kinds of organisms clearly form a coherent subtopic within the ontology.

Iteration 3 The third iteration already produces a partition of the remaining concepts into islands that are all of the required size thereby ending the iteration. Figure 6 shows the result of the third iteration that contains the following additional modules:

- Entity
- Organization
- Device
- Anatomical Structure
- Fully Formed Anatomical Structure
- Substance
- Organic Chemical

Most of these modules make sense, the only questionable results are the separation of fully formed anatomical structures from anatomical structures and the separation of organic chemical from substance. We will analyze and discuss these results in more details in the conclusions.

IV. CONCLUSIONS

We proposed a practical approach for ontology partitioning that works on the structure of the ontology and can be completely automated. Looking at the result of the

example application we get a first idea about the strengths and weaknesses of the algorithm. We can see that the algorithm generates some modules that meet our intuition about the nature of a module quite well. In some cases subtrees that could be considered to form one module are further split even if the complete subtree does not exceed the upper size limit. This can be explained by an unbalanced modelling of the ontology as subtrees tend to be split up at concepts with a high number of direct subclasses compared to its sibling classes. This phenomenon often reflect a special importance of the respective concept in the ontology that also justifies the decision to create a separate model for this concept. The iterative strategy frees us from determining a lower bound for the size of modules. As a result, however, the algorithm sometimes create rather small modules. In our example the manufactured objects module for example only contains four concepts. This normally happens when the root concept of a small subtree is linked to a concept that has many direct subclasses. For the result of the partitioning method these subsets are often pathological because coherent topic are split up into a number of small modules that do not really constitute a sensible model on their own.

V. References

[1] V. Batagelj. Analysis of large networks - islands. Presented at Dagstuhl seminar 03361: Algorithmic Aspects of Large and Complex Networks, August/September 2003.

[2] V. Batagelj and A. Mrvar. Pajek - analysis and visualization of large networks. In M. Jnger and P. Mutzel, editors, *Graph Drawing Software*, pages 77–103. Springer, 2003.

[3] P. Bouquet, F. Giunchiglia, F. van Harmelen, L. Serafini, and H. Stuckenschmidt. C-owl: Contextualizing ontologies. In *Proceedings of the 2nd International Semantic Web Conference ISWC'03*, Lecture Notes in Computer Science, pages 164–179, Sanibal Island, Florida, 2003. Springer Verlag.

[4] R.S. Burt. *Structural Holes. The Social Structure of Competition.* Harvard University Press, 1992.

[5] Jennifer Golbeck, Gilberto Fragoso, Frank Hartel, Jim Hendler, Jim Oberthaler, and Bijan Parsia. The national cancer institute's thesaurus and ontology. *Journal of Web Semantics*, 1(1), 2003.

[6] H. Stuckenschmidt and M. Klein. Integrity and change in modular ontologies. In *Proceedings of the International Joint Conference on Artificial Intelligence - IJCAI'03*, pages 900–905, Acapulco, Mexico, 2003. Morgan Kaufmann.

[7] The Gene Ontology Consortium. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.

[8] R. Volz, A. Maedche, and D. Oberle. Towards a modularized semantic web. In *Proceedings of the ECAI'02 Workshop on Ontologies and Semantic Interoperability*, 2002.

[9] J. Wielemaker, G. Schreiber, and B. Wielinga. Prolog-based infrastructure for RDF: performance and scalability. In D. Fensel, K. Sycara, and J. Mylopoulos, editors, *The Semantic Web - Proceedings ISWC'03, Sanibel Island, Florida*, pages 644–658, Berlin, Germany, october 2003. Springer Verlag. LNCS 2870.

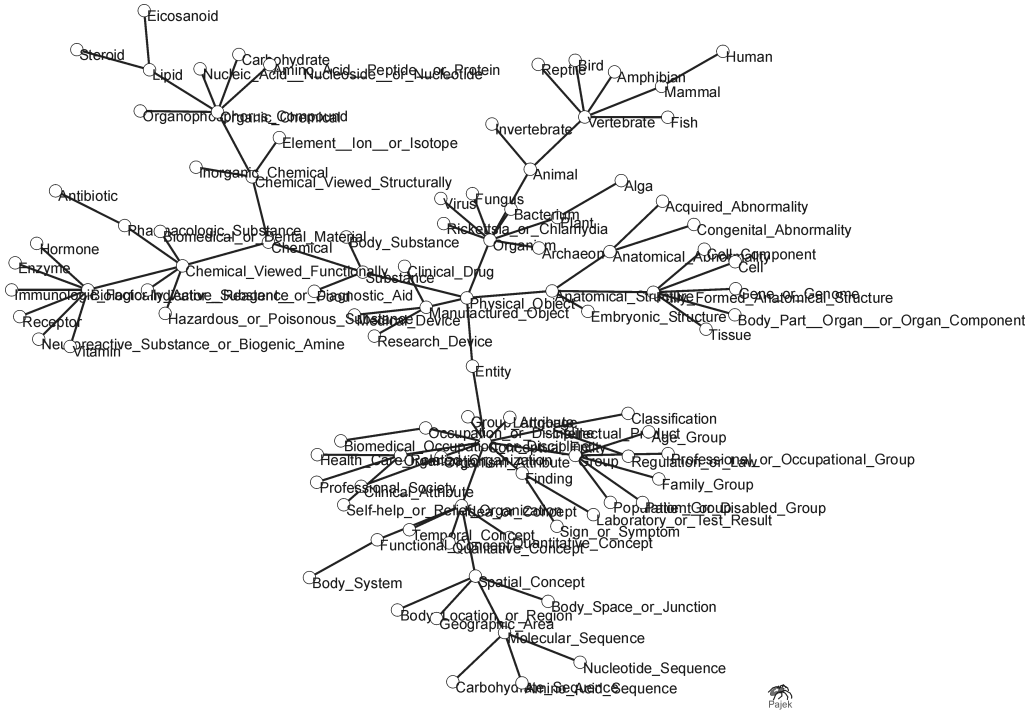

Fig. 1: Class hierarchy graph for the entity-related part of the UMLS semantic network

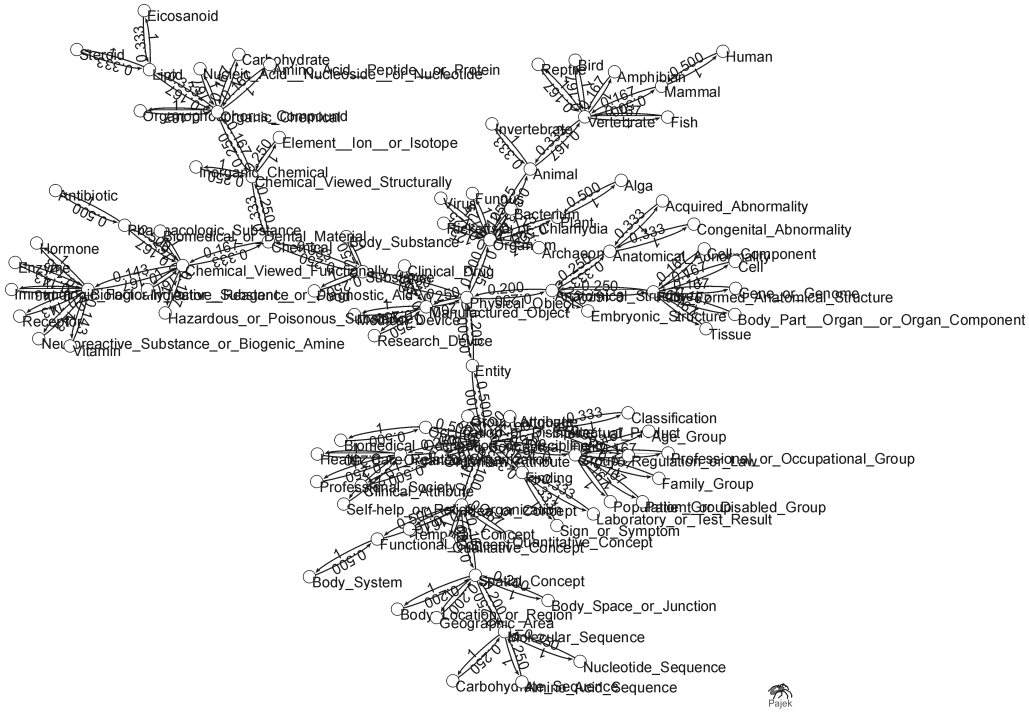

Fig. 2: Relative Strength Networks for the Class Hierarchy Graph

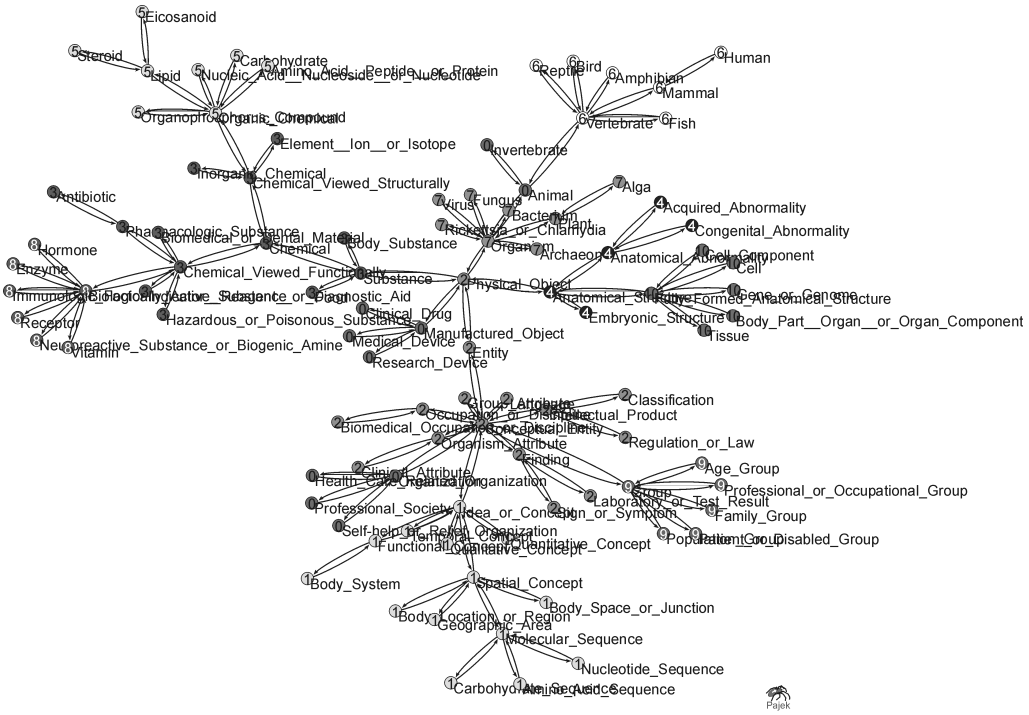

Fig. 3: Islands of size between 5 and 15 (10 Islands, 6 unassigned nodes)

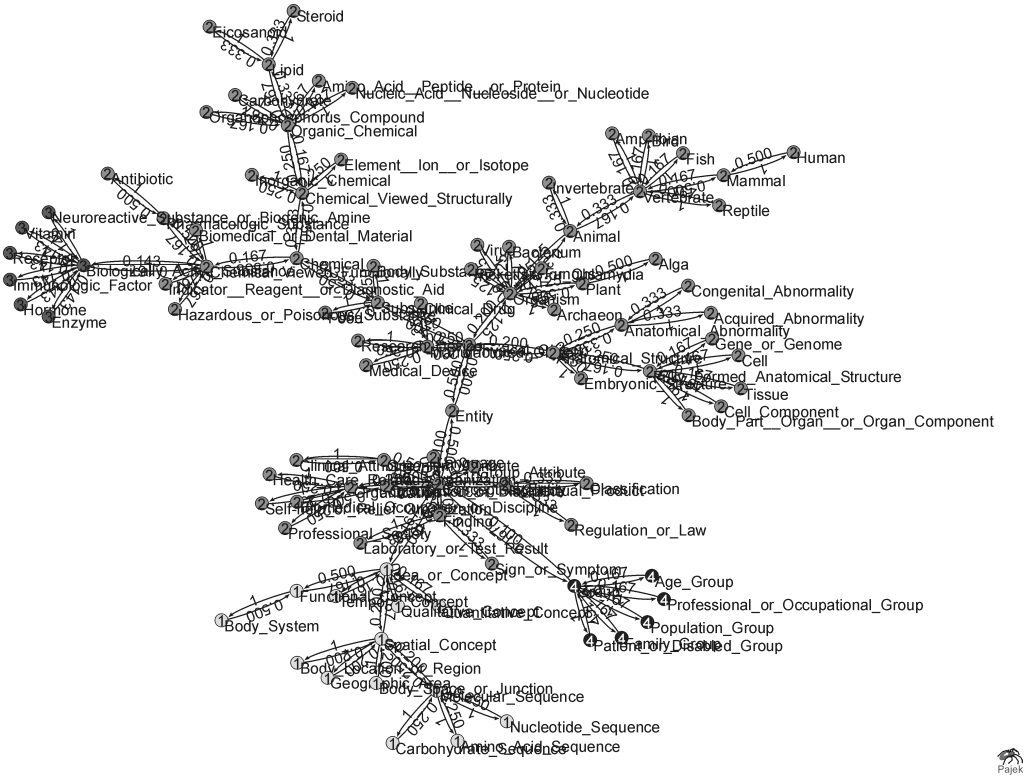

Fig. 4: Result of the first Iteration

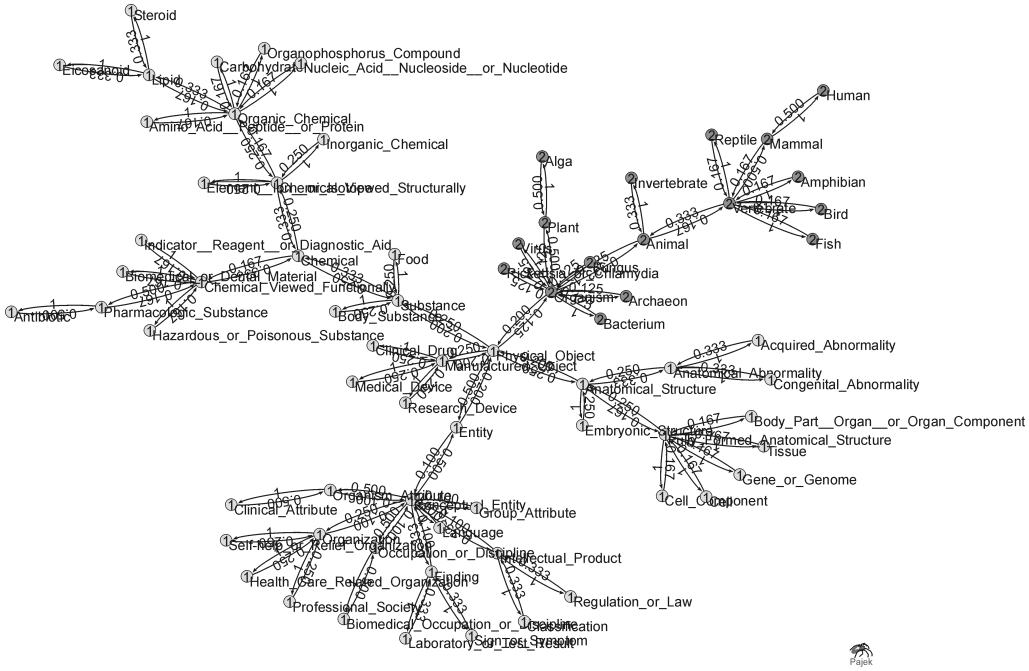

Fig. 5: Result of the second Iteration

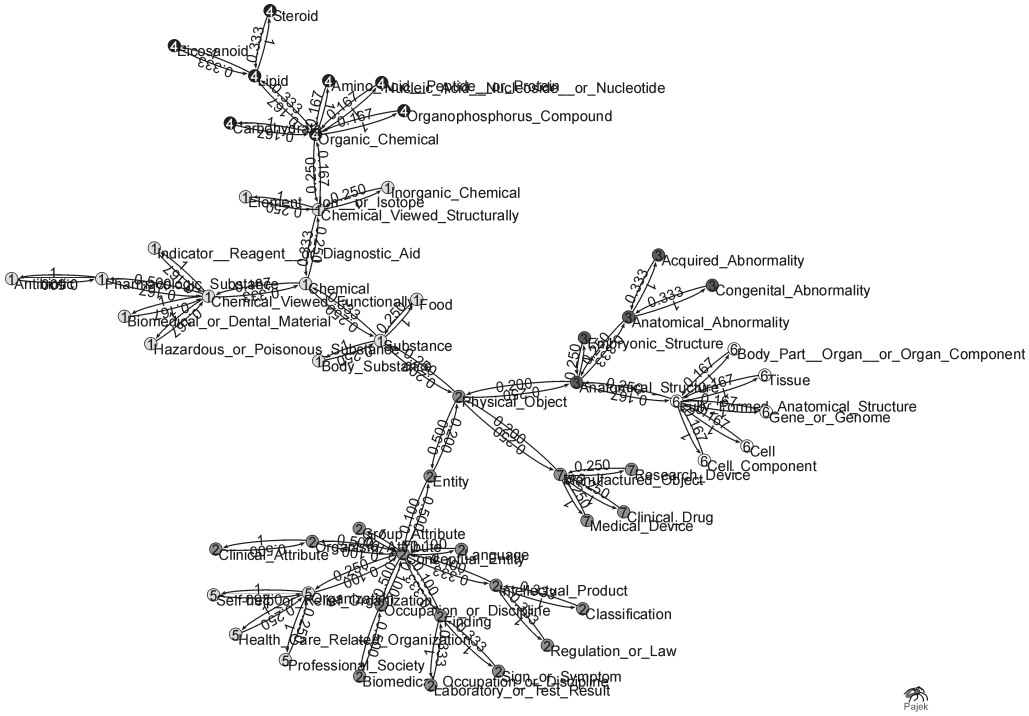

Fig. 6: Result of the third Iteration